4

THE STATE UNIVERSITY OF NEW JERSEY
RUTGERS

RUTGERS UNIVERSITY
Center for Expert Systems Research

Quarterly Report:
Empirical Analysis and Refinement of Expert System Knowledge Bases

Contract Number

TECHNICAL REPORT

Department of
Computer Science

Laboratory for
Computer Science
Research

4

RUTGERS UNIVERSITY
Center for Expert Systems Research

Quarterly Report: *Empirical Analysis and Refinement of Expert System Knowledge Bases*

Contract Number
N00014-87-K-0398
Office of Naval Research

November 30, 1988

Principal Investigators:
Sholom M. Weiss
Casimir A. Kulikowski

1. Technical Project Summary

Knowledge base refinement is the modification of an existing expert system knowledge base with the goals of localizing specific weaknesses in a knowledge base and improving an expert system's performance. Systems that automate some aspects of knowledge base refinement can have a significant impact on the related problems of knowledge base acquisition, maintenance, verification, and learning from experience. The SEEK system was the first expert system framework to integrate large-scale performance information into all phases of knowledge base development and to provide automatic information about rule refinement. A recently developed successor system, SEEK2 [Ginsberg, Weiss, and Politakis 88] significantly expands the scope of the original system in terms of generality and automated capabilities. The investigators expect to make significant progress in automating empirical expert system techniques for knowledge acquisition, knowledge base refinement, maintenance, and verification.

2. Principal Expected Innovations

The investigators will demonstrate a rule refinement system in an application of the diagnosis of complex equipment failure: computer network troubleshooting. The expert system should demonstrate the following advanced capabilities:

- automatic localization of knowledge base weaknesses
- automatic repair (refinement) of poorly performing rules
- automatic verification of new knowledge base rules
- automatic learning capabilities

3. Objectives for FY89

These are our objectives for the current year, Fiscal year 89:

- full demonstration of refinement system, using subset of DEC's Network Troubleshooting Consultant (NTC). System will automatically recover from many forms of damage to knowledge base.
- full demonstration of system with capabilities for automatic refinement, and verification of knowledge base consistency. Empirical experiments will be performed and results will be reported.
- demonstration of significant automated rule learning capabilities.
- demonstration of extended system capabilities for alternative control strategies and representations.

- completed comparative studies of empirical techniques for machine learning, statistical pattern recognition, and neural nets.

4. Summary of Progress

During the previous year the following was accomplished:

- initial functioning equipment diagnosis and repair knowledge base, suitable for refinement. This is a subset of DEC's Network Troubleshooting Consultant (NTC).

- initial demonstration of functioning equipment diagnostic system with capabilities of localization of weak rules, automatic refinement, automatic verification.

- demonstration of initial rule learning capabilities.

- development of case generation simulator and randomized rule modifier.

- initial comparative studies demonstrating superiority of PVO rule induction procedure.

This work is the basis for further progress in developing an automated refinement system. We are pursuing the refinement and learning tasks from both an expert system rule-based perspective and a machine learning rule induction perspective. In order to develop the strongest form of refinement system, we have examined numerous techniques for empirical rule induction. We have also developed a procedure, Predictive Value Maximization, that shows strong results for induction of single relatively short rules. Our fundamental objective is to mix the best rule induction procedures with a rule-based expert system to achieve the strongest empirical results.

Here are the highlights of progress in meeting our stated objectives for the first quarter of fiscal year 89:

- We have completed an extensive empirical comparison of machine learning rule induction techniques with statistical pattern recognition techniques, and neural nets. Four real-world data sets were analyzed using different techniques. The study required over 6 months of Sun 4 CPU time. The results should be reported in the next quarterly report.

- We have begun adding a procedure to the refinement system that uses rule induction techniques. These results should be available during the next quarter.

- We have developed models for measuring and comparing the results of rule-based expert systems. A preliminary report is attached to this quarterly report.

5. Financial Review

1. Basic contract dollar amount: \$536,919 (9/1/87-8/31/89)

2. Dollar amounts and purposes of options: None

3. Total spending authority received to date: \$475,000 through 1/31/89

4. Total spending to date: \$225,526 through 11/30/88

5. Monthly expenditure rate: We have charged very little in salaries over the first academic year in anticipation of funding larger amounts in summer salaries (when we are able to devote major efforts to the research project). We have also brought on board one more graduate assistant to assist in this research, resulting in higher salary expenditures anticipated throughout the current (1988-89) academic year.

6. We have funded a total of approximately \$225,526 to date. This would, therefore, result in an average monthly expenditure rate of \$15,035.

7. Major non-salary expenditures planned within this increment of funding: None

8. Date next increment of funds is needed: January, 1989.

I. Technical Report

A preliminary technical report, *Models for Measuring Expert System Performance,* is enclosed with this quarterly report.

References

[Ginsberg, Weiss, and Politakis 88]
Ginsberg, A., Weiss, S., and Politakis, P.
Automatic Knowledge Base Refinement for Classification Systems.
Artificial Intelligence :197-226, 1988.

Models for Measuring Expert System Performance

Nitin Indurkhya *and* Sholom M. Weiss
Department of Computer Science, Rutgers University, New Brunswick, NJ 08903

Abstract [illegible]

Scoring schemes for measuring expert system performance are reviewed. Rule-based classification systems and their error rates on sample data are considered. We present several models of measurement that are categorized by four characteristics – mutual exclusivity of classes, unique answers provided by the system, known correct conclusions for each case and use of confidence factors to weight the system's conclusions. An underlying model of performance measurement is critical in determining which scoring strategy is appropriate for a system and whether a comparison of different systems can be made. (KF)

1 Introduction

Although many expert systems have been developed, relatively few have been formally evaluated [8, 12, 19]. Different methods of measuring performance can be used for each system, making it difficult to compare their performance. In this report several underlying models of measuring performance are considered. Here *performance* refers to the error rates of systems on sample data.

A survey of strategies for measuring performance of expert systems can be found in [12]. Rather than surveying performance measurement strategies in existing systems, the focus, in this paper, is on underlying models. Classification type rule-based expert systems [2, 18] are considered, although the results can be carried over to other classification systems [7]. The discussion is confined to scoring and measuring performance, as opposed to techniques (such as *train and test*) for estimating bias in the measured error rates [4]. A review of performance measurement models is critical in determining which scoring strategy is appropriate for a system and whether a comparison of different systems can be made.

2 Measurement Models For Classifiers

A performance measurement model for a classification expert system requires several basic elements. Figure 1 illustrates the components of a standard performance measurement model. In Figure 1, the scoring of a case depends on the system's conclusion and the expert's conclusion. The structure of the classifier is not critical – it could be a set of rules, a decision tree or even a procedural algorithm. The only thing that is of concern is that the classifier takes case-data as input and outputs an answer. Thus, for evaluation purposes, the expert classifier is treated as a black box. In

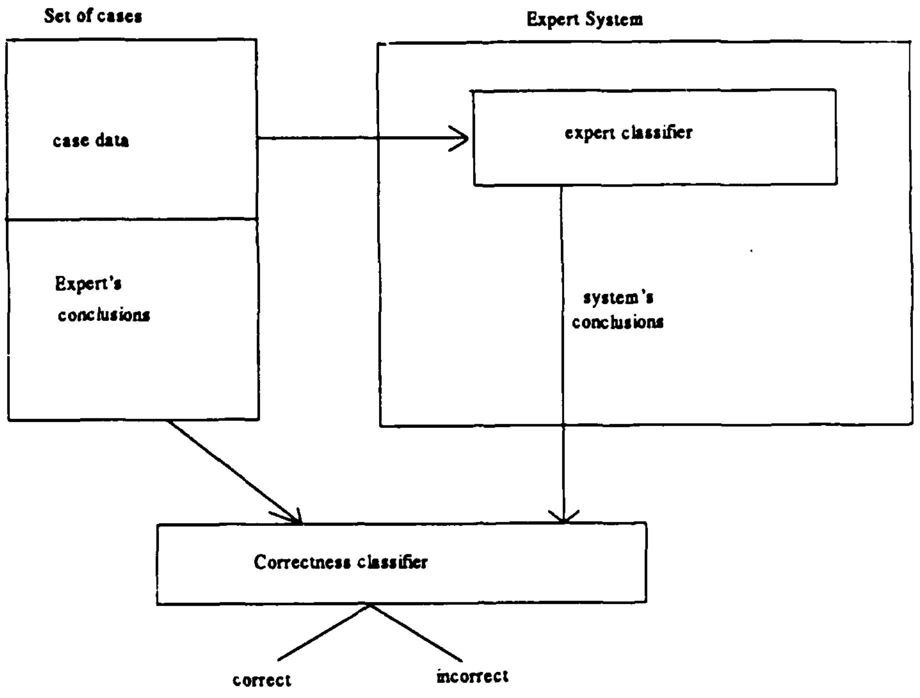

Figure 1: Measuring Expert System Performance

fact the view taken is that of an examiner who is given a system and told to measure its performance. The examiner does not care how the system is designed internally and focuses on the output of the system for a given set of input values.

For the purposes of measuring performance, the following characteristics are of interest:

Unique Answer This attribute refers to whether the system's conclusion for a case is unique. Thus, if a classifier always outputs a single class as its conclusion for all the test cases, the system is said to be *UA restricted.* Otherwise, the system is *Not-UA restricted.* As an example, consider the following conclusions of two systems classifying among (*class1*, *class2*, *class3*) over the same set of cases:

Case	System A	System B
case1	*class1*	*class1*, *class2*
case2	*class3*	*class2*
case3	*class1*	*class1*, *class3*

For the given set of cases, System A is *UA restricted* and System B is *Not-UA restricted.* This will affect what kind of scoring strategy is best suited for evaluating these systems. The scoring scheme for System A will be different from that for System B.

Mutual Exclusive Classes This attribute refers to whether, for a given case, only one class can be satisfied. This will be true if the classes considered are mutually exclusive. It is assumed that the classes span the space of alternatives[1]. Thus, if every case can be classified in one and only one of the classes, then we say that the domain is *ME restricted.* If there are cases that can be classified in more than one class, then we say that the domain is *Not-ME restricted.*

Known Case Conclusion This attribute refers to whether the correct conclusion for a case is *known* and *undisputed.* The correct conclusion can be obtained from an expert or the true conclusion can be obtained by observation over time until the event is over. Thus, for a system which predicts rain for the next day, the true conclusion can be obtained by waiting until the next day and observing whether it rains or not. This will be the correct conclusion for the case. If the true conclusion is obtainable, then it is also *undisputed.* However, if the correct conclusion is obtained from an expert, then it may be disputable. For a case, two different experts may diagnose differently.

Weighted Answers This attribute refers to whether the system's answers are classified by uncertainty factors. If the system gives unweighted answers without any certainty factors, its answers are considered definite. For example, given a three class situation of *class1*, *class2* and *class3*, an expert system with unweighted answers might give the following output – (*class2*, *class3*). This will be taken to mean that the case can be classified as both *class2* and *class3* and that both the classifications are equally definite. A system gives weighted answers if it makes use of an uncertainty model in its reasoning. Such a system typically provides a likelihood measure or a Confidence Factor (CF) with each class – the CF reflecting how strongly the system supports this class as an answer. For example, for the three class situation described above, an expert system with weighted answers might have the following output:

class1	0.7
class2	0.5
class3	0.0

Thus, the system has no confidence that the case can be classified as *class3* and has a higher confidence in classifying the case as *class1* than as *class2*. In the above example, note that the confidence factors do not sum up to 1.0. This means that the uncertainty model is not based on probability theory. This is not unusual. Many systems use non-probabilistic uncertainty models. For purposes of measuring performance, the theory underlying the uncertainty model is not significant.

The above features will be used to categorize scoring strategies and describe models for measuring performance.

[1] If the classes do not span a complete set of possibilities, then an additional class *None-of the-above* can be added to the list of classes.

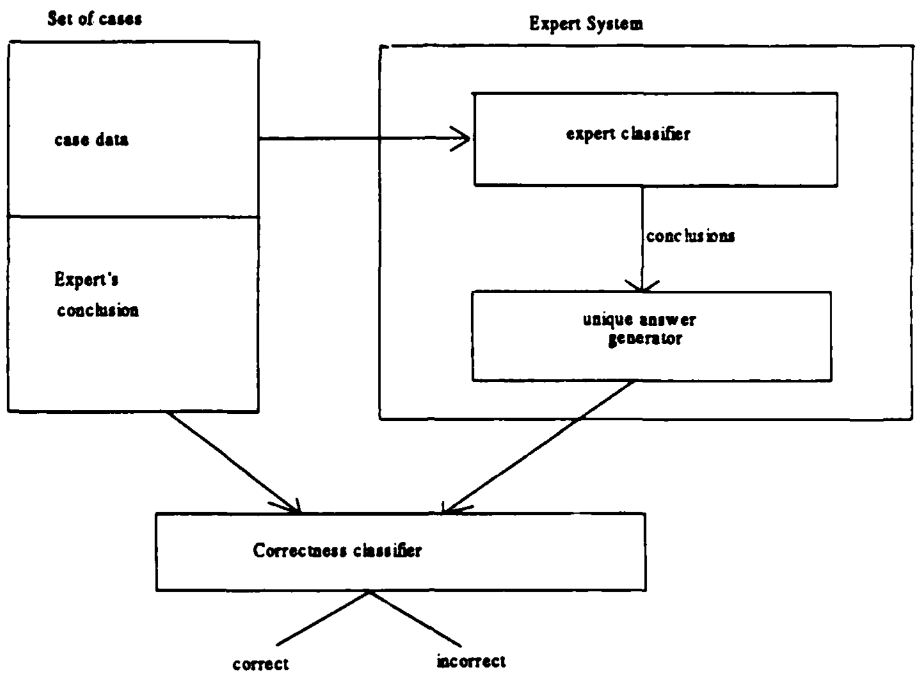

Figure 2: Systems Restricted To Unique Answers And Mutual Exclusivity

3 Models For Systems Restricted To Mutual-Exclusivity And Unique Answers

The simplest situation occurs when the correct conclusion for the case is ***undisputed*** and ***known***, the classes are mutually exclusive and span the possibilities, and the system's conclusion is unique. A classifier for a ME-restricted domain, usually gives a unique answer for any input data[2]. Thus for a ME-restricted problem, most classifiers would be UA-restricted and for any input data would give a unique answer. In a rule-based system, where no rules or multiple rules can be satisfied, a ***unique answer*** is equivalent to conflict resolution in which a single answer is selected by some criteria such as highest confidence factor, etc. These criteria can be encapsulated into a ***unique answer generator***. Because of this, it does not matter whether the answers are weighted or not. The situation for UA-restricted classifiers in ME-restricted domains is shown is Figure 2. Performance measurement models for such systems are examined in the rest of this section.

3.1 The Correctness Model

The most common measurement strategies are based on measuring the total number of correct decisions made by a system. This measure is popular among machine learning systems [9] where

[2]When classifiers for ME-restricted domains are not UA-restricted, this leads to analytical difficulties. This issue will be discussed later on in this report.

performance improvement is demonstrated by counting the number of correct responses before and after learning. The underlying model is one in which the number of correct/incorrect responses are being noted. Thus, for example, for a two class situation (mutually exclusive classes):

$$\text{System's response} = C_{system}$$
$$\text{Correct response} = C_{correct}$$

A case is classified as correct or incorrect using the following table:

	C_{system}	
$C_{correct}$	Class1	Class2
Class1	Correct	Incorrect
Class2	Incorrect	Correct

Using this information, the following table is filled up:

Correct	Incorrect

The metric of percentage of errors is calculated as:

$$\% \text{ of errors} = \frac{Incorrect}{Correct + Incorrect} \times 100.$$

Alternatively, one could use a ***confusion matrix*** [7]. A confusion matrix is a N×N table of ***actual class*** against ***classified class*** where N is the number of classes. This is illustrated by the following example. Suppose for a three class situation, a classifier produces the following confusion matrix in scoring a set of 50 cases:

	C_{system}		
$C_{correct}$	Class1	Class2	Class3
Class1	13	2	5
Class2	2	14	1
Class3	3	1	9

Note that each entry in the confusion matrix refers to the number of cases having the corresponding C_{system} and $C_{correct}$. Thus, for example, the entry in the first row and second column means that in 2 cases C_{system} was Class2 and $C_{correct}$ was Class1. Correctly classified cases fall on the diagonal moving from the upper left to the lower right. Based on the the confusion matrix, the overall error rate can be determined:

Correct	Incorrect
36	14

This gives the following result for the system:

$$\% \text{ of errors} = \frac{14}{50} \times 100 = 28\%$$

This model is the simplest measurement model and compares performance at a coarse level. It does not distinguish between different types of errors.

3.2 The Positive/Negative Correctness Model

Some systems use a more detailed model than the correctness model described above. Instead of classifying a case as correct, it is now classified as *True Positive* or *True Negative.* Instead of classifying a case as incorrect, it is now classified as *False Positive* or *False Negative.* This classification is done with respect to a class.

As an example, the tables involved for a two class situation are shown below. Note that the tables are with respect to class1:

	C_{system}	
$C_{correct}$	Class1	Class2
Class1	True Positive	False Negative
Class2	False Positive	True Negative

For N classes, the table with respect to Class1 would be as follows:

	C_{system}				
$C_{correct}$	Class1	Class2	Class3	...	ClassN
Class1	TP	FN	FN	...	FN
Class2	FP	TN	FN	...	FN
Class3	FP	FN	TN	...	FN
⋮	⋮	⋮	⋮	⋱	⋮
ClassN	FP	FN	FN	...	TN

Using the above classification for each case, the following table is calculated for each class:

	T+	T-
H+	TP	FN
H-	FP	TN

For each table, H+ and H- add up to the total number of cases as do T+ and T-. The tables for the classes are related. A FP case for *class1* will be a FN case for *class2.* The sum of all the TP's and FN's from all the classes equals the total number of cases. Since FN's for a class are FP's for other classes, it is enough to report TP and FP values for each class.

A number of metrics are possible using this model. Some of them are:

Sensitivity	TP/H+
Specificity	TN/H-
Predictive value(+)	TP/T+
Predictive value(-)	TN/T-
Accuracy	(TP+TN)/TOTAL

A system which uses this model is described in [5]. The metrics *specificity* and *sensitivity* are commonly used in scoring laboratory tests in medicine. It is important to note that there are two kinds of errors being distinguished: *false positive* errors and *false negative* errors. They could be considered of unequal importance.

The computations can be done easily using a *confusion matrix* similar to the example given for the correctness model. The tables described above can provide useful information for improving the performance of the classifier. For example, it might indicate that the classifier does very well on all the classes but one – thereby indicating that the performance can be improved by trying to improve the results for this one class. SEEK2 [6] is an example of a system that uses the matrix of True Positives, False Negatives, False Positives and True Negatives as a source of information for improving performance which it measures using the correctness model.

3.2.1 The Correctness And Cost Model

This is a generalization of the positive/negative correctness model. There is now a cost factor associated with making a FP or FN judgment. This kind of model is useful when it is known that the cost of making a FP decision is different from making a FN decision. For example, in a situation of approving credit card requests, the cost of a FN decision (not approving a good applicant) is relatively low (the annual card fee and interest payments) while the cost of a FP decision (approving an applicant who is unable to pay his bills) is high (the average annual purchases of a credit card holder). Thus one calculates the following table:

	T+	T-
H+	TP	FN × FNcost
H-	FP × FPcost	TN

The positive/negative correctness model is a special case of this model with $FNcost = FPcost = 1$. Note that the cost function need not be linear. For example, if it is important that the number of FN's be not more than 50% of the FP's, then the non-linear cost function which needs to be calculated would be $f(FP, FN) = FP/FN$. The same set of metrics as used with the positive/negative correctness model are relevant here.

A number of optimality measures for classifiers can be understood as special cases of this model. Thus, the criteria described in [7, Pg 63] – *Minimum cost, Minimax error* and *Fixed error rate* can be all seen to be within the scope of this model. The reader is referred to [7] for details.

4 Models For Systems Restricted To Mutual-Exclusivity And Non-Unique Answers

In all the models discussed so far, three important assumptions have been made:

Known Case Conclusion: The correct conclusion was *known* and *undisputed.*

ME Restricted Domain: The classes were mutually exclusive.

UA Restricted Classifier: The expert system incorporated a unique answer generator thereby ensuring that for any input data, the system always gave a unique answer.

These assumptions simplify considerably the factors affecting measurement. We now relax the third of the abovementioned constraints and consider *Not-UA restricted* expert systems. So the situation is that the classes are mutually exclusive and the classifier can give more than one class as its conclusion[3].

It was mentioned earlier that when a classifier for a ME restricted domain is not UA restricted, this leads to analytical difficulties. The reason for this is that having non-unique answers when only one of the classes can be a correct answer leads to difficulties in evaluation. The strategies for dealing with these problems can be categorized on the basis of whether the answers are unweighted or weighted[4].

4.1 Models For Systems With Unweighted Answers

In this situation non-unique answers arise because rules for more than one class are satisfied and the system does not have a strategy for picking among the classes. Sometimes, this indicates that the system is not properly constructed. For the purposes of evaluating such a system, the system designers must provide schemes for picking one of the multiple answers as the answer to be used in scoring the system. In effect, the system must be augmented with methods which act as filters to the system output and make it UA-restricted. A good example of this can be found in AQ15 [11]. In reporting performance of AQ15, Michalski and his colleagues describe the use of special routines of the kind mentioned above which force the system to be UA-restricted.

There are situations, however, when one could measure performance in the presence of multiple answers. For example, let there be 100 classes and for a particular case, suppose system A narrows its diagnosis down to 3 classes while system B narrows its diagnosis down to 20 classes. Let the correct conclusion be among the multiple answers of system A as well as those of system B. Clearly, system A has done better than system B. Such an evaluation is reported by Spackman [16] who compares the performance of his learning system CRLS with AQ15. Refer to Section 5.1.2 where the technique is discussed. However, when there are only two mutually exclusive classes, accepting multiple answers is awkward.

4.2 Models For Systems With Weighted Answers

When the system provides weighted answers, this can give rise to non-unique answers if the system itself does not select one of the classes as a conclusion. There are two main strategies for dealing with this situation.

[3]The conclusion for each case is still known and undisputed.

[4]Because the systems being considered are Not-UA restricted, confidence factors do make a difference now, as compared to the previous section where UA restricted systems were considered.

4.2.1 Using Models Restricted To Mutual-Exclusivity And Unique Answers

Given the standard interpretation of confidence factors and given the knowledge that the classes are mutually exclusive, for purposes of scoring the system, the class in which the system has highest confidence can be picked as the C_{system}[5]. This makes the system *UA-restricted* and so the ME/UA restricted models can be used to score the system. For example, if a system gives the following output for a case:

class1	0.7
class2	0.5
class3	0.0

Then, for the purposes of scoring the system over the case, *class1* will be picked as C_{system}.

4.2.2 Measurement By Averaging Weights

By picking the most certain answer, in effect, the uncertainty in the answers is being ignored. A situation where the correct conclusion has second-highest confidence may be better than a situation where the correct conclusion has fourth-highest confidence. By picking the class with highest confidence, both the above situations are marked as wrong and classified equivalently. This may not be desirable. The solution is to use techniques which measure partial correctness in these situations by averaging over the weights. Two such techniques are described below:

Measurement Using Distance Metrics

One way of measuring partial correctness is to use a closeness measure rather than forcing the system to choose between correct and incorrect classification. For N diagnoses, consider an n-dimensional space. Each axis is the confidence factor of a diagnosis (hence it is scaled from 0 to 1). The expert's conclusion is a point in this space and so is the system's conclusion. Consider an appropriate distance metric between these two points in this space. The mean or variation over all cases could be taken as a performance metric.

Thus, for example, consider the case of the earlier section. Let $C_{correct}$ for the case be *class2*. Since there are 3 classes, points in 3-dimensional space need be considered. Let *class1* lie along the *x-axis*, *class2* along the *y-axis* and *class3* along the *z-axis*. Then, $C_{correct}$ can be represented by the point *(0,1,0)*. Similarly, C_{system} is the point *(0.7,0.5,0.0)*. If the squared distance is taken as a metric, then the case is assigned a score of 0.74.[6] Once scores are calculated for a number of cases, the average squared distance can be reported as a performance metric. This kind of performance measurement using a distance metric is popular among neural net systems [15].

[5] There is the problem of how to handle *ties* – more than one class having the highest confidence, but this complication shall be ignored for the moment.

[6] The smaller this score, the better the performance on the case.

Measurement Using ROC Curves

This measurement strategy, described in [17], is useful in the two class situation. An ROC curve is a plot of sensitivity against 1-specificity. The values of specificity and sensitivity are calculated for various levels of decision confidence of the system. These levels are chosen along the confidence range. The area under the ROC curve is calculated as an performance metric. A 100% accurate system having 100% sensitivity and 0% specificity, has an area of 1.0. This measure is independent of the occurrence of class members in the test data as well as of the decision bias of the system.

The strategy can be best illustrated by an example. Consider the following data obtained from an expert system that classifies patients as having Rheumatoid Arthritis (RA) or not. 121 cases are classified by the system in four different categories. These categories are obtained by dividing the confidence range into four intervals and classifying cases into these four categories instead of the original two categories (RA or not-RA)[7]. This gives us the results of columns 2 and 3.

Rating category	$C_{correct} = RA$	$C_{correct} \neq RA$	*Sensitivity*	$1 - Specificity$
definitely RA	21	4	0.50	0.05
probably RA	10	5	0.74	0.11
possibly RA	7	13	0.90	0.28
definitely not RA	4	57	1.0	1.0
total	42	79		

The sensitivity and specificity are calculated by the "rating procedure". This involves viewing each rating category as representing a threshold. All cases in higher categories are viewed as positive decisions. Thus, viewing ***Possibly RA*** as representing a threshold, we compute the following:

$$\begin{aligned}
\text{Number of true positive decisions} &= 21 + 10 + 7 = 38 \\
\text{Total number of positive decisions} &= 42 \\
\text{Number of false positive decisions} &= 4 + 5 + 13 = 22 \\
\text{Total number of negative decisions} &= 79 \\
\text{Sensitivity} = \text{True Positive decisions/Total Positive decisions} &= 0.90 \\
\text{1 - Specificity} = \text{False Positive decisions/Total negative decisions} &= 0.28
\end{aligned}$$

The other values in columns 4 and 5 are calculated similarly. Thus rating information is used as though decision thresholds were being used instead. Once we the results of columns 4 and 5, we now have enough information to plot five points. The resulting ROC curve is shown in Figure 3 with an area of 0.87.

5 Relaxing Mutual Exclusivity Of Classes

In the previous section, it was shown that relaxing the UA restriction while keeping the ME restriction leads to problems. Now the ME restriction is relaxed while still keeping the constraint of having a known undisputed conclusion for each case.

[7] The choice of four categories is arbitrary. It could have been anything else.

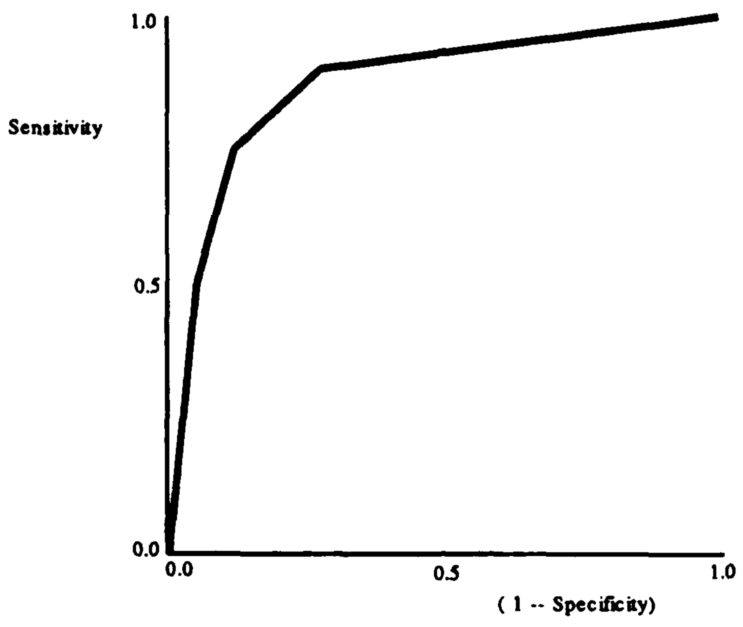

Figure 3: An ROC Curve

If the classes are not mutually exclusive, multiple classes may apply to the same case data. For example, a patient may have a bad throat, fever, diabetes and lung cancer at the same time. Any system that diagnoses this set of health disorders cannot make assumptions about mutual exclusivity of these classes. Thus, from Figure 2, the unique answer generator can be removed. The new situation is depicted in Figure 4.

It is immediately clear that a large part of performance measurement now rests with the correctness classifier. In the earlier models, since there was only one expert conclusion and one system conclusion for each case, the correctness classification was trivial – just check to see if expert's conclusion was equal to the system's conclusion or not. Now, however, the problem is complicated because there are multiple classes for the conclusions of both the expert system and the expert. How does one score a case for which the expert system and the expert agree of some conclusions and disagree on some others ? Any model that deals with measuring performance of such systems must necessarily address this problem and have a scheme for resolving it.

5.1 Models For Systems With Unweighted Answers

As before, the simplest situation is considered first. Assume that the correct conclusions are known and undisputed and that the system's conclusions are unweighted answers Note that $C_{correct}$ and C_{system} are sets of classes. Thus for a case one could have $C_{correct}$ = (*class1*, *class3*) and C_{system} = (*class1*, *class2*). The models described below are proposed as possible ways of scoring such systems.

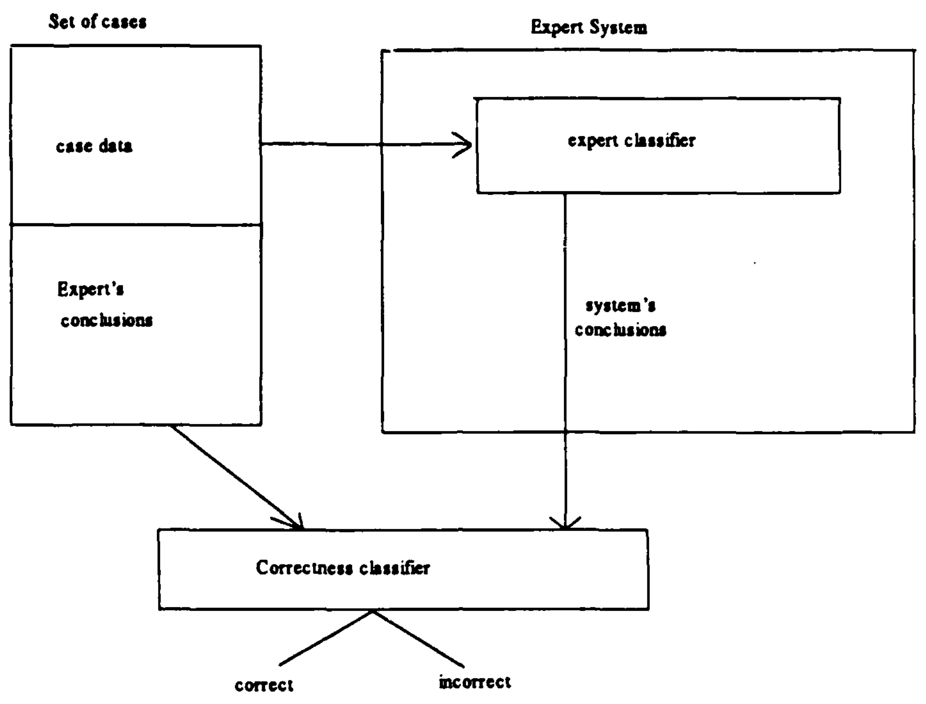

Figure 4: Measurement With Multiple Diagnoses

.1 The Case Correctness Model

The simplest measure of performance is to measure how many *cases* are incorrect. The definition of an *incorrect* case may not be domain-independent. Consider the following examples:

1. If the domain is medicine and we are dealing with a set of diseases all of which are very expensive to test and at the same time all the diseases are life threatening, then for a case in question, the system must match the expert exactly in its conclusions. If it guesses more disease classes than the expert, then the further testing required to weed out the false positives would make the system not very cost effective. On the other hand, if the system does not guess all the diseases, then the patient will not have much confidence in the system. These criteria translate into the following correctness classifier:

   If $C_{correct}$ and C_{system} match exactly, then the case is correct or else it is incorrect.

2. Again consider a set of diseases which are extremely expensive to test for precisely. Assume that an expert system is being installed for checking the presence of these diseases during regular medical examinations. For the system to be effective and useful, the system must not raise false alarms which would necessitate the expensive testing. For such a situation, the following correctness criteria would be appropriate:

   If C_{system} is a subset of $C_{correct}$ then the case is correct or else it is incorrect.

Given that the correctness classifier classifies each case as correct or incorrect (possibly by some domain-dependent scheme), the percentage of errors can be computed as a measure of the system performance. The metric of percentage of errors is calculated as before:

$$\% \text{ of errors} = \frac{Incorrect}{Correct + Incorrect} \times 100.$$

In many situations it is often important to determine the degree of error. Otherwise one cannot distinguish between the performance of systems such as follows:

	Conclusions	
System	Case A	Case B
Expert	*class1*, *class2*	*class2*, *class3*
System M	*class2*	*class1*, *class2*, *class3*
System N	*class3*	*class1*

Assume the correctness criteria is exact match. Although, both systems M and N got the cases wrong, it seems, intuitively, that System M did better than System N. This is not reflected at all in the measurement by case correctness.

5.1.2 The Partial Correctness Model

It was seen above that the case correctness model was relatively coarse. Another way of measuring correctness would be to employ some kind of closeness measure rather than to force the system to choose between correct and incorrect classification. This can be achieved by allowing partial correctness of cases. The following example illustrates this strategy. Suppose that a system reports the following performance on two cases:

Case	$C_{correct}$	C_{system}
Case1	*class1*, *class2*	*class2*, *class3*
Case2	*class2*, *class3*	*class1*, *class2*, *class3*

Thus for Case1, $C_{correct}$ contained 2 classes and the system got 1 of them right, so it is given a score of 1/2 on correctness. While this handles the case of partial correctness on cases, it is inadequate unless it is coupled with some measure on how precise the system was. One such measure is the positive predictive value[8]. Thus, for the above example, we get the following two measures:

$$\text{Accuracy} = \frac{1+2}{2+2} = 0.75$$
$$\text{Predictive Value } (+) = \frac{1+2}{2+3} = 0.60$$

These metrics were in fact used in [11] to measure performance of AQ15 for cases with multiple diagnoses.

[8]The ratio of the number of classes correctly identified by the total number of classes in C_{system}.

5.1.3 The Diagnostic Performance Model

The case correctness model does not specify a system's performance relative to a class. This may be important for a rule-based system, for example, to determine the performance of the rules for a particular class (so that they can be refined). One way of measuring performance by diagnostic class is as follows:

Split each case into *p-cases* corresponding to the number of correct classes for that case. Thus, each *p-case* has one conclusion from the $C_{correct}$'s for the case which generated it. This is the $C_{correct}$ of the *p-case*. Now we measure performance for each *p-case*. The system classifies a *p-case* as correct if the $C_{correct}$ of the *p-case* is among the system's conclusions. The notions of False Positive and True Negative can be defined similarly.

Thus for example, consider a domain where there are three classes: *class1, class2* and *class3*. Consider the following two cases:

case number	$C_{correct}$	C_{system}
1	class1, class2	class1, class3
2	class1, class3	class2, class3

The following performance measurement by diagnostic class is obtained:

class	TP/H+	False Positives
class1	1/2	0
class2	0/1	1
class3	1/1	1
total	2/4	2

What has been done, effectively, is that the system is being viewed as a ME/UA restricted system. For each *p-case*, each of the C_{system}'s is the system's answer and is either equal to the one and only one $C_{correct}$ for that *p-case* or not. Thus, all the metrics discussed earlier for such systems are applicable here. Costs can be attached to the various false positive and false negative errors of each class in order to get a metric which measures performance more accurately. SEEK2 [6] uses this model for handling cases with multiple diagnoses.

The drawback of this model is that it cannot be used to estimate the performance of the system over *cases*. If one tries to do so, the performance measures will be biased towards cases with more multiple conclusions.

5.2 Measurement Models For Handling Weighted Answers

So far non-ME restricted systems without uncertainty have been examined. This simplifies the interpretation of system output. Any class that gets fired by the system rules, is a part of C_{system}. In this section scoring of systems with weighted answers for non-mutually exclusive classes is considered. In an earlier section, the effects of adding an uncertainty model to systems classifying among mutually exclusive classes were analyzed. The main problem had been in defining what should be considered as C_{system}. The same problem is encountered when the classes are not mutually exclusive. It is assumed that the expert's conclusions are undisputed and definite, and the system produces an ordered list of classes (ordered by some measure like confidence factor CF). The CF attached to a class reflects the system's belief that the case can be classified in the class. For example, consider a case for which the expert's conclusions are *(class1, class2)*. Let the system produce the following list of classes ordered by confidence factors: *((class1, 0.9) (class3, 0.7) (class2, 0.4) (class4, 0.3))*. Leaving aside $C_{correct}$ for the moment, the difficulty is in assigning an interpretation to C_{system}. There are several strategies that can be used here, each giving rise to a different scoring scheme:

1. One strategy might be to ignore certainty factors for performance measurement. Thus, for the above example, one would determine C_{system} = (class1, class2, class3, class4). All classes with non-zero CF's would be considered as the system's answers. This strategy ignores the reasoning under uncertainty employed by the system and is thus not a very precise measure of performance.

2. Another alternative would be to ask the domain expert to define what an answer should be in terms of the confidence factors of the system. This would depend on how the confidence factors are being interpreted by the domain experts. Thus, one could decide to treat every class with a CF greater than 0.5 as a part of C_{system}. For the example, we get C_{system} = ((class1, 0.9) (class3, 0.7)). Having used the CF's to determine what should be the answer, they are ignored for scoring the system response. In general, there can be no domain-independent way of using CF's to fix thresholds for answers because the confidence factors have different interpretations in different domains. Thus, by this strategy, the system can use uncertainty models for reasoning but is obligated to also provide a mechanism for determining acceptable answers for performance measurement.

3. Another strategy might be to thrust the responsibility of interpreting the system's output to the correctness classifier. Thus the correctness classifier looks at the system output (which is a list of classes with confidence factors) and the $C_{correct}$ (which is a set of classes) and determines whether the system's output is acceptable as an answer or not. One way the correctness classifier might do so is to count the number of classes in $C_{correct}$ and accept an equal number of classes from the system's output as C_{system} and then do the performance measurement using any of the schemes described for the situation when C_{system} is definite. Thus, for the example, since $C_{correct}$ contains two classes, the two classes with highest CF would be assumed to be the system's answer and C_{system} would be taken to be (class1, class3). This is the strategy used by SEEK2 in scoring cases with multiple diagnoses for the AI/RHEUM knowledge base which uses an uncertainty model.

4. Another way of measuring correctness would be some kind of closeness measure rather than to force the system to choose between correct and incorrect classification. This has been

discussed earlier for systems with mutual exclusivity and non-unique answers. The technique is applicable even when the classes are not mutually exclusive.

6 Measurement When True Case Conclusion Not Known

In the preceding sections, techniques for measuring performance were described for conclusions that are undisputed and known. In some domains the answer for each case is not truely known. The expert himself may be wrong and so his conclusion may not be the true conclusion. Also, two expert may give differing conclusions for the same case data. The problem here is how does one find the *gold standard*, the conclusion against which to compare performance of the system. Several methods for solving this problem can been used:

Find the true conclusion This means that the expert opinion is ignored and an attempt is made to get the real answer by observing real-world events. For example, if one is building a system which forecasts the next day's weather, then the true conclusion can be found by waiting until the next day and observing the weather. As another example, consider a system trying to determine life expectancy of cancer patients.

Compare with human expert's performance Sometimes it may not be possible to obtain the true conclusion. In such situations, the strategy used is to measure the performance by comparing with the performance of human experts.

This strategy was used to validate a version of MYCIN in the domain of meningitis [19]. For each of ten cases, six treatment recommendations were obtained – one was obtained using MYCIN, one was the actual therapy used in treating the patient and the other four were obtained from experts at four different experience levels. For each case, the six recommendations were randomly ordered and presented to eight outside experts. Each outside expert gave his own recommendation and rated the six recommendations as identical to his own, acceptable or unacceptable. A majority of the evaluators approved MYCIN's choice 70% of the time. None of the other four experts achieved better approval than MYCIN.

Allow experts to attach confidence factors What if the expert refuses to commit himself ? Or if he cannot commit himself ? The strategy here is to allow the expert to attach confidence factors to his conclusions. This strategy is particularly useful if the performance data is not very extensive, and the expert attaches confidence factors which are known to be compatible with the confidence ratings assigned by the system.[9] For each case, one obtains the conclusions from the expert with confidence factors for each class. For each class the signed difference between the CF value of system and the expert is computed. For a case, the average difference is computed over all classes to obtain a metric of performance over a case. Alternatively, the

[9]This would be the situation if a model of expert reasoning was being accurately designed and it was known that the expert can give indefinite answers. So the uncertainty model being used would have to conform to the uncertainty model used by the expert himself.

average difference could be computed for each class over all cases to obtain a metric of performance with respect to each class. Thus, for example, consider the following data for a case:

class	$CF_{correct}$	CF_{system}	Difference
class1	0.80	0.61	0.19
class2	0.40	0.50	-0.10
class3	0.30	0.30	0.00

The average difference between the confidence values for the above data is 0.03 which is 3% of the 0 to 1 scale. Such a performance measurement was reported for PROSPECTOR [3].

7 Conclusion

This paper has explored the various issues involved in measuring expert system performance on the basis of errors made on sample case data. The simplest scenario was considered first: when the conclusion for each case is known and undisputed, the classes are mutually exclusive and the system generates a single answer. These assumptions were then relaxed one by one and various approaches used to the handle the problems arising therein were analyzed. Most of them have been shown to be efforts to make the assumptions applicable again. From the point of view of performance measurement, four features of expert systems were identified as interesting: (1) Unique Answer, (2) Mutual Exclusivity, (3) True Conclusion Known and (4) Weighted Answers. The categorization of measurement models has been done over these four features. In the table below, some expert stems have been classified on the basis of these features:

System	UA	ME	TCK	WA
INTERNIST-1 [13]	*n*	*n*	*y*	*y*
MYCIN [19]	*n*	*n*	*n*	*y*
PROSPECTOR [3]	*n*	*n*	*n*	*y*
AI/RHEUM [8]	*n*	*n*	*y*	*y*
AQ15 [11]	*y*	*y*	*y*	*n*
CRLS [16]	*n*	*y*	*y*	*n*
NetTALK [10]	*y*	*y*	*y*	*y*
Decision Trees [1, 14]	*y*	*y*	*y*	*y*
Classical Pattern Recognition [4]	*y*	*y*	*y*	*y*

In the table, ME, UA and TCK and WA refer to the four features. A *y* refers to the presence of the feature and a *n* refers to its absence. Some combinations of these features are awkward. Thus, it is awkward to consider systems where the classes are not mutually exclusive but the system always gives a unique answer. Note that some of the systems mentioned in the table have been designed to handle cases from more than one category. This is because some categories are broader than others. Thus, systems which handle cases with non-unique answers, can also handle cases with unique answers. Systems have been classified into the broader category that covers the cases over which measurement is likely to be done. Besides specific systems, two classes of systems have been categorized. The classical pattern recognition algorithms referred to are ***supervised learning*** algorithms. Note that classical pattern recognition algorithms and neural net algorithms tend to follow the simplest model of measurement. Expert systems, on the other hand, tend to diverge from the simplest model. Consequently, they need a more complex representation and are more difficult to evaluate.

Given the above categorization of classifiers, one can more readily determine which set of measurement models are appropriate for a given classification system. Determining the underlying performance measurement model is critical in determining which scoring strategy is appropriate.

References

[1] Breiman, L. et al. (1984). *Classification and Regression Trees*, Wadsworth, Inc., Belmont, California.

[2] Clancey, W. (1984). *Classification Problem Solving*, Proc. of AAAI-84. Pgs 49-55.

[3] Duda, R.O. et al. (1979). *Development of the Prospector Consultation System for Mineral Exploration*, Final Report SRI Project 6415, SRI International, Menlo Park, California.

a, R.O. and Hart, P.E. (1973). *Pattern Classification and Scene Analysis*, John Wiley and Sons, Inc., New York.

[5] Galen, R. and Gambino, S. (1975). *Beyond Normality: The Predictive Value and Efficiency of Medical Diagnoses*, John Wiley and Sons, Inc., New York.

[6] Ginsberg, A., Weiss, S.M. and Politakis, P. (1988). *Automatic Knowledge Base Refinement for Classification Systems*, Artificial Intelligence 35 (1988). Pgs 197-226.

[7] James, M. (1985). *Classification Algorithms*, John Wiley and Sons, Inc., New York.

[8] Kingsland III, L.C. (1985). *The Evaluation of Medical Expert Systems: Experience with The AI/RHEUM Knowledge-based Consultant System in Rheumatology*. Proc. Ninth Annual Symposium on Computer Applications in Medical Care. Pgs 292-295. IEEE Computer Society Press, Washington D.C.

[9] Laird, J. (ed). *Proc. of the fifth International Conference on Machine Learning 1988*, Ann Arbor, Michigan.

[10] Sejnowski, T.J. and Rosenberg, C.R. (1987). *Parallel Networks that Learn to Pronounce English Text*, Complex Systems 1 (1987). Pgs 145-168.

[11] Michalski, R.S., Mozetic, I., Hong, J. and Lavrac,N. (1986). *The AQ15 inductive learning system: An overview and experiments.* Tech. Rept. ISG 86-20, Dept. of Computer Science, Univ. of Illinois at Urbana-Champaign, Urbana, Illinois.

[12] Miller, P.L. (1986). *The evaluation of artificial intelligence systems in medicine,* Computer Methods and Programs in Biomedicine 22 (1986). Pgs 5-11.

[13] Miller, R.A., Pople, H.E. and Meyers, J.D. (1982). *INTERNIST-1, An experimental computer-based diagnostic consultant for general internal medicine.* N. Engl. J. Med. 1982. Pgs 307-468.

[14] Quinlan, J.R. (1986). *Induction of Decision Trees,* Machine Learning 1 (1986). Pgs 81-106.

[15] Rumelhart, D.E., Hinton, G.E. and Williams, R.J. (1986). *Learning Internal Representations by Error Propagation* in D.E. Rumelhart and J.L. McClelland (Eds.), "Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. 1: Foundations". MIT Press.

[16] Spackman, K. (1988). *Learning Categorical Decision Criteria in Biomedical Domains.* Proc. of the fifth International Conference on Machine Learning. Pgs 36-46. Univ. of Michigan, Ann Arbor, Michigan.

[17] Swets, J.A. (1988). *Measuring the Accuracy of Diagnostic Systems.* Science (3) Jun 1988. pgs 5-1293.

Weiss, S.M. and Kulikowski, C. (1984). *A Practical Guide to Designing Expert Systems.* Rowman and Allanheld, Totowa, New Jersey.

[19] Yu, V.L. et. al. (1979). *Antimicrobial selection by a computer: A blinded evaluation by infectious disease experts.* J. Amer. Med. Assoc. 242 (1979). Pgs 1279-1282.